REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1998	Technical Report

4. TITLE AND SUBTITLE
Tradeoff Between Dwell Time and Frequency of Revisits of Sensors

5. FUNDING NUMBERS

6. AUTHOR(S)
Prof. Don Barr

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USMA Operations Research Center
West Point, New York 10996

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A.
Approved for public Release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Department of Defense has expressed heightened interest in managing information processes related to combat operations. This has highlighted the analytic community's need for information-related analysis methods, which begs the question of how to measure combat information. One measure, called "information gain", has been developed and applied at the US Military Academy in recent years. Information gain is based on modeling a tactical commander's uncertainty about his adversary's state (location of enemy units, for example) using probability distributions are "updated", using Bayes' formula and target movement characteristics. Information gain is defined to be the decrease in Shannon's entropy from the prior to the posterior situations. An impression can be gained of the growth in potential "situation awareness" of a commander by plotting cumulative information gain over time, as data from various sources are received.

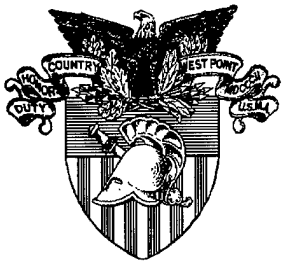
An issue related to the design architecture of certain sensor systems is the tradeoff between sensor dwell time and frequency of revisits of the sensor. We have implemented information gain analysis in a simple model of sensor coverage and target movement, and have examined the cumulative information gain over a sequence of time steps. The model is implemented in a simulation written in Visual Basic. It allows examination of the effects of changes in a variety of parameters of interest. We have examined effects on information gain of varying dwell times under a number of conditions. One results is an indication that larger dwell times may have an advantage over more frequent visits when targets are quite mobile.

14. SUBJECT TERMS
Tradeoffs Between Dwell Time and Frequency of Revisits of Sensors

15. NUMBER OF PAGES
34

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	



United States Military Academy
West Point, New York 10996

TRADEOFFS BETWEEN DWELL TIME AND FREQUENCY OF REVISITS OF SENSORS

DEPT. OF SYSTEMS ENGINEERING TECHNICAL REPORT

By

Donald R. Barr, Professor
Department of Systems Engineering
US Military Academy
West Point, NY 10996

(914) 938-4374
fd4168@usma.edu

August 1998

EXECUTIVE SUMMARY

The Department of Defense has expressed heightened interest in managing information processes related to combat operations. This has highlighted the analytic community's need for information-related analysis methods, which begs the question of how to measure combat information. One measure, called "information gain," has been developed and applied at the US Military Academy in recent years. Information gain is based on modeling a tactical commander's uncertainty about his adversary's state (location of enemy units, for example) using probability distributions over the set of possible states. When data are received from an information source the probability distributions are "updated," using Bayes' formula and target movement characteristics. Information gain is defined to be the decrease in Shannon's entropy from the prior to the posterior situations. An impression can be gained of the growth in potential "situation awareness" of a commander by plotting cumulative information gain over time, as data from various sources are received.

An issue related to the design architecture of certain sensor systems is the tradeoff between sensor dwell time and frequency of revisits of the sensor. We have implemented information gain analysis in a simple model of sensor coverage and target movement, and have examined the cumulative information gain over a sequence of time steps. The model is implemented in a simulation written in Visual Basic. It allows examination of the effects of changes in a variety of parameters of interest. We have examined effects on information gain of varying dwell times under a number of conditions. One result is an indication that larger dwell times may have an advantage over more frequent visits when targets are quite mobile.

The author is indebted to the National Reconnaissance Office for support of this project.

TABLE OF CONTENTS

INTRODUCTION

As a result of studies concerning the architecture of the US Army's future force, known as "Century XXI" and "Army After Next," it has become apparent that information dominance is a critical requirement (TRADOC Pamphlet 525-5 1994). Thus there is heightened interest in optimizing battlefield information systems and managing related information processes (Sovereign 1996). It appears critically important to include measures of combat information in designing and evaluating combat systems and devising tactics for their use. Intuitively, most people believe information has value in combat, but it is not obvious how the underlying relationships might be quantified. Commonly used analytic measures are based on system throughput characteristics such as the volume or rate of messages, message quality or timeliness, system reliability, or characteristics of the data given in the messages, such as detection rates (MacWillie 1992). At the other end of the spectrum are the extraction of meaning from data received and its use in decision-making. The cognition of, and response to, information conveyed in a given set of data depends upon the receiving commander. This human process depends on the circumstances of the situation, as well as the personality, training, and experience of the commander.

In recent years, we have developed a model of information gain at a level between dealing with characteristics of the physical communications system and dealing with human cognition and response of the decision-maker (Barr and Sherrill 1995, Sherrill and Barr 1997). We approach measuring the level of information a commander possesses at a given point in time by modeling the amount of *uncertainty* he faces, in terms of probability distributions over sets of possible enemy states. When the commander receives information about his adversary, the probability distributions are updated, using Bayes' formula or other means, to reflect the new state of the commander's uncertainty. The information gained as a result of the data received is measured by the decrease in Shannon's entropy from the prior to posterior distributions. This can be viewed as providing an upper bound on the increase in actual "situation awareness" of the commander, because the latter depends on his cognitive abilities to process the new information and fuse it with all available information. This approach is not *ad hoc*; in (Barr and Sherrill 1996) we show that, under seemingly reasonable assumptions, decrease

in entropy is the unique appropriate measure of information gain. The measure has been successfully applied in a variety of systems evaluations (Sherrill and Barr 1996, Marin and Barr 1997).

The work reported here is focused on determining information gains related to receipt of data about target locations in a given area of concern. We model the search process as a sequence of glimpses in a row of cells that partition the area of concern. The search pattern is cyclic. It starts at the "left" end of the row of cells, then progresses sequentially through adjacent cells on the "right." When the right end of the row is reached, search resumes at the left end, and the process is repeated. Dwell is modeled as a selected number of successive glimpses in each given cell before moving to search the adjacent cell. "Tracking" of a detected target during a dwell period is modeled by not allowing the target to move during the portion of a dwell period following its detection. Thus the sensor maintains line of sight (LOS) with a detected target during dwell periods. LOS is lost when the sensor moves to the adjacent cell. A detected target may be re-detected during a dwell period, strengthening the location information about that target. If a target is not present in a cell being scanned by the sensor, a "detection" may be declared, forming a false alarm. The sensor is assumed to have known operating characteristics (probabilities of detection and false alarm) for each search in each cell. Outcomes of sensor glimpses are independent, except for the case of tracking detected targets during dwell periods.

The foregoing model is a crude representation of the following hypothetical situation. The area of concern is a valley whose width is less than the sensor footprint diameter. Cells are formed along the valley by successive non-overlapping sensor footprints. An unmanned aerial vehicle may travel up the valley, then orbit back to the starting point following a path outside the valley. This forms a sensor without dwell capability. A ground observer on a high point above the valley scans each cell for a given amount of time, following the same general cyclic pattern. This represents a sensor with possible dwell times. If the ground observer detects a target, he continues to track it for a time equal to the remainder of dwell period from the time of detection.

Our goals in this report are as follows:

- To describe the basis of the information gain model (Section 1);

- to document a Visual Basic program we developed to examine information gains under various conditions (Section 2 and the Appendix); and
- to present some tradeoff indications from our model (Section 3).

Because of the low level of resolution of our model, the tradeoff results we report are merely indicative of broad interactions among parameters such as dwell time and target movement probability. They suggest the *existence* of important tradeoffs, but only give very rough guidance to the conditions where break-even points occur for actual sensor systems in a real search environment.

1. THE INFORMATION GAIN MEASURE

In this section, we introduce the information gain measure and discuss some of its properties. We also describe the random walk we used to model target motion.

A slight extension of Shannon's development of entropy in a communications framework (Shannon 1948) provides a characterization of the information gain measure. Suppose a "Blue" commander's area of concern consists of a set of non-overlapping cells that may contain an enemy asset. Suppose $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ is the prior probability distribution over n possible states, representing the Blue commander's uncertainty of Red's presence or location at some specific time, and suppose the uncertainty he has at some later time is represented by a posterior distribution, $\mathbf{p}^*$. Denote the information gained in resolving the uncertainty represented by $\mathbf{p}$ to that represented by $\mathbf{p}^*$, by $\delta(\mathbf{p},\mathbf{p}^*)$. Under several reasonable assumptions about the properties of the function δ, it follows the function must be of the form

$$\delta(\mathbf{p},\mathbf{p}^*) = \Sigma p^*_i \ln(p^*_i) - \Sigma p_i \ln(p_i) , \qquad (1)$$

which is just the decrease in Shannon's entropy from the prior to posterior situations. A formal statement of this result is given in (Barr and Sherrill 1996), along with some elementary properties of the function δ.

If a discrete system can be in state i with probability p_i; $i=1,2,\ldots,n$, Shannon defined its *entropy* to be $-\Sigma p_i \ln(p_i)$, where the sum is over all n states and the logarithm is to the base 2, so entropy is measured in *bits*. (Since zero is not in the domain of the logarithm function, we *define* $0 \cdot \ln(0)$ to be 0). Entropy is a measure of the dispersion of probability mass over points, *without regard to what those points are.* This distinguishes

entropy from common statistical measures of dispersion such as variance and inter-quartile range. If a system can be in any of n possible states, the entropy of the system can range between 0 (when the exact state of the system is known) and $ln(n)$ bits (when the state of the system is uniformly distributed over the possible states).

Information gain does not depend on the labels used for outcomes in the sample space. This is entirely reasonable in our applications, because the labels of cells and the coordinate system of the battle area are inventions of the analyst; they are not inherently relevant to the gain in information about target location, for example. A simple interpretation of values of information gain can be based on the elimination of possible states by receipt of data. Suppose initially any of n states are equally likely, and data are received showing that (n – m) of the states are not possible. The posterior would then be uniform over m states, so the information gain would be ln(n/m) bits. For example, if the area of a "region of uncertainty" is halved by data from a reconnaissance report, the information gain is one bit. In general, an information gain of n bits is equivalent to the gain realized in reducing an area of uncertainty to $1/2^n$ its original size.

The posterior distribution may be affected by the possible movements of targets, in addition to the receipt of data from sensors. In this report, we assume each target follows a simple symmetric random walk with reflecting barriers at each end of the row of cells. In each time cycle, a target remains stationary with probability q, and moves to the adjacent cell in either direction with probability (1-q)/2. At the end cells, the target can move only in the direction of its adjacent cell, with probability 1-q. This is an ergodic Markov chain, and its properties are well known (Goodman, 1988). For example, if there are a total of nc cells, the limiting distribution of target location is given by the mass function

(1/2(nc-1), 1/(nc-1), 1/(nc-1), …, 1/(nc-1), 1/2(nc-1)).

If it is assumed the tactical commander knows of this target behavior, it is reasonable to use this distribution as the initial prior. In this case, the initial entropy is $\log_2(nc-1) + 1/(nc-1)$.

If a given target is not being tracked, the posterior distribution of its location, computed with receipt of data from the sensor, should be "decayed" to account for possible movement of the target. This is done using the one-step transition probabilities

for the Markov chain. Thus, location information following detection of a target in a certain cell, which provides relatively high probability the target is located in that cell, will be reduced in subsequent periods when line of sight (LOS) is lost, because the target may have moved from the cell. In particular, if **p** is the vector representing a given target's location mass function at a particular time when LOS has been lost, then the decay of information in the following single time step is represented by the location mass function **p*** after one step of the Markov chain. Letting nc denote the number of cells, **p*** is given by $\mathbf{p}^* = \mathbf{p} \cdot \mathbf{P}$, where **P** is the nc x nc one-step transition matrix,

$$P = \begin{pmatrix} q & 1-q & 0 & 0 & 0 & \dots & 0 \\ \frac{1-q}{2} & q & \frac{1-q}{2} & 0 & 0 & \dots & 0 \\ 0 & \frac{1-q}{2} & q & \frac{1-q}{2} & 0 & \dots & 0 \\ & & & \cdot & & & \\ & & & \cdot & & & \\ & & & \cdot & & & \\ 0 & 0 & \dots & 0 & \frac{1-q}{2} & q & \frac{1-q}{2} \\ 0 & 0 & \dots & 0 & 0 & 1-q & q \end{pmatrix}$$

Target locations are assumed to be independent, so the total information gain over a given time step is the sum of the gains for the individual targets. As described above, the gain for each target is the result of sensor data regarding this target (consisting of either "target not seen" or "target detected") and possible target movement during the time interval. Sensor data are incorporated using Bayes' formula, and movement effects are incorporated using the random walk equations.

2. DESCRIPTION OF THE SIMULATION MODEL

In this section we describe and document the program we developed to facilitate evaluating information gain with the model discussed in Section 1.

Since the sensor may generate random false alarms and the targets may move from cell to cell, it appears difficult to derive closed-form analytical expressions for expected information gains achieved as searching progresses. We have taken the approach of simulating sensor and target positions, and sensor results, over time. Time is measured in terms of a count of "searches" representing time intervals required for the sensors to complete one glimpse of a cell. The simulation is implemented in Visual Basic. The program simulates locations of targets with random starting locations and random walk behavior after each search (with no movement when a target is being tracked during a dwell period). The starting locations are generated independently, in accordance with the prior distribution specified by the user. The random walk is governed by the probability the target moves, set by the user. The program keeps track of which cell is being glimpsed at each search, and whether each target is in that cell. If a target is in the cell being glimpsed, there is a detection of the target by the sensor with probability specified by the user for this cell. If a target is not in the cell being glimpsed, there is a false "detection" of the target with probability specified by the user for this cell. Otherwise, there is no detection of the target for the cell (which is also information that is taken into account in computing information gain).

The logic sequence of the program is as follows:

a. input parameters for the search, including
   iter = number of iterations of the simulation,
   nc = number of cells in the region of interest,
   nt = number of targets moving amongst the cells of the region,
   P(0,J) = prior distribution of locations of targets,
   pd(J) = probability of detection in each cell J,
   pf(J) = probability of false alarm in each cell J,
   ns = number of searches per iteration,
   dc = dwell time of sensor, and
   1-q = probability target moves after each glimpse (when not being tracked);
b. draw initial target locations, using the specified prior distribution;
c. set the cell pointer (the cell now being searched);

d. conduct a search of the chosen cell and update the location distribution for each target in accordance with data resulting from the search;
e. decay the location distribution of each target to account for possible target movements;
f. compute current entropy of the location distributions, and update statistical counters;
g. move targets in accordance with random walk probabilities;
h. if fewer than ns searches have been completed for this iteration, goto step b above;
i. output summary statistics related to information gain, ask the user if another run is desired: if so, goto step a; if not stop.

There are three uses of the random walk model of target locations in this analysis.

- It enables the simulation to determine whether a claimed "detection" is an actual detection or a false alarm, depending on whether there is LOS with the target involved. LOS with a target exists when the sensor is looking in the cell containing the target. False alarms are associated with random targets for which there is no LOS, at a rate for each such target that makes the over-all false alarm rate in each search equal to the probability of false alarms specified by the user.
- It provides the mechanism for decay of the location distribution of each target not being tracked, to account for possible movements of such targets.
- The limiting Markov chain distribution provides a logical alternative to the assumption of a uniform prior distribution (or other prior distribution) for target locations. The prior is used to generate initial target positions, and to compute the initial entropy in the run.

Input of parameters to the simulation and output of information gain statistics from the simulation use a link with an Excel workbook prepared in advance by the user. The path to the Excel file must be specified by the user on the Visual Basic (VB) input form displayed at run time (see Figure 1). All of the parameters, in their simplest form, can be input directly when the program is run, on the VB input form displayed. If the user wishes to assign different probabilities of detection of targets in different cells, then the nc-dimensional vector of probabilities must be placed in the Excel file read by the program, and the pd box in the VB input form set to blank. Similarly, if the user wishes

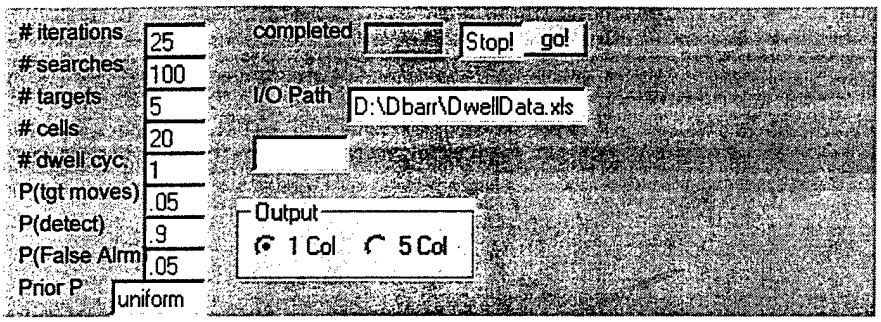


Figure 1. The VB input form. The user can set parameter values prior to each run. Default values are shown in the figure. To prepare for the first run, set the parameter values and click "go." For subsequent runs, click the "Another?" button (shown as a blank box in the figure), set parameters and click "go." The "completed" box indicates a running tally of iterations completed as a run is being conducted.

to vary the probability of false alarm over cells, the vector of false alarm probabilities must be input via the Excel file and the pf box in the VB input set to blank. A similar scheme is used to input prior location distributions other than "uniform" and "limit" (the limiting distribution of the random walks of the targets). To choose uniform or limit prior distributions, the user should enter "uniform" or "limit," respectively, in the prior box of the VB input form.

A "run" of the program consists of a simulation of "iter" iterations of "ns" searches for each of "nt" targets in "nc" cells. When the program has completed a run, and written data out to the Excel file as described below, the user is given an option to set new parameter values and make another run. This facilitates examination of the sensitivity of information gain to variations in parameters of interest. For example, many of the plots shown in the next section, showing comparisons of performance with five values of dwell time, were generated by selecting "Another" [run] when prompted. If the user clicks the "Another" box on the VB input form, he/she is prompted to set new parameter values (in our example, dwell time) and click "go" to produce another run with the new parameter values. The output to the Excel workbook is arranged to accommodate many runs, without overwriting output values. The user can then open the Excel file and analyze the output, make comparative plots, etc.

To accommodate use of modular arithmetic for computing cell locations in case of "wrap around" at the end of the row of cells, the cells are numbered in the range [0, nc-1]. To input "custom" values for any of the three nc-dimensional vectors described in the preceding paragraph, they must be placed in Sheet1 of the Excel workbook, in the following cells:

	cells in search area		
	0	1 ...	(nc-1)
parameter	cells of Excel Sheet1		
detection probability	C3, D3, ...,		X3
false alarm probability	C4, D4, ...,		X4
prior distribution	C5, D5, ...,		X5

If detection probability is the same for all cells, the common value can be entered in the pd box of the VB input form, and the vector need not be present in the Excel Sheet1. A similar comment holds for false alarm probabilities. If the user enters a prior distribution as "uniform" or "limit" in the VB form, the prior vector need not be present in the Excel Sheet1. If any of these vectors are input from the VB input form, the program writes the vectors to the Excel Sheet1 in the locations specified above, to provide documentation of the input values for the run. When several runs are made in one session, only the most recent vectors are written to Sheet1. All other parameter values for each run are also recorded in the Excel Sheet1.

Output from one run of the program consists of either one column vector or five column vectors of dimension ns = (number of searches), written to the Excel Sheet1, depending on the choice of the user (see Figure 1). The columns begin in row 15 of Sheet1, and extend to row 15 + ns - 1. Descriptive headers for the columns are written to Sheet 1. If the user specifies 5-column output, the output is identified as follows:

Column A: average (over iterations) of total entropy of the targets at each search;

Column B: standard deviation of the total entropy at each search;

Column C: average information gain from each search;

Column D: average cumulative information gain at each search; and

Column E: average cumulative information index (the information gain divided by the initial entropy) at each search.

An example of output for a run with 5-column specification is shown in Table 1. The output in Sheet1 has been truncated to the range B1..I18 to make the table fit the page. On Sheet1, the three top rows extend to the right to give numerical values for nc = 10 cells, and the data columns extend downward to row 254 to cover values from each of the ns = 240 searches. Figure 2 below shows a plot of the five columns of data, to give an idea of the range of values obtained for each statistic reported. Note the standard deviation of entropy is nearly constant, at about 1.0 in this run. Note also that "steady state" is reached in this case after about 30 searches, which seems reasonable in light of the fact that three complete sweeps of the area of concern (10 cells) have been completed at that time.

P(detect)	0.9	0.9	0.9	0.9	0.9	0.9	. . .
P(false)	0.05	0.05	0.05	0.05	0.05	0.05	. . .
Prior	0.1	0.1	0.1	0.1	0.1	0.1	. . .
iter=25							
ns = 240							
nt = 3							
nc = 10							
P(Mv)0.05							
dc = 1							
run# 1							
Entropy Mean	Entropy std dev	Info Gain	Cum Gain	Cum Gain Index			
9.173022	0.806427	0.792762	0.792762	0.079548			
8.535983	1.062078	0.63704	1.429801	0.143471			
7.79889	1.066822	0.737092	2.166894	0.217433			
6.853439	0.953436	0.945451	3.112345	0.312303			
. . .	. . .	. . .	. . .	. . .			

Table 1. Portion of output from a 5-column run, written to Excel Sheet1. Note that all parameter values for the run are written, documenting conditions for the run.

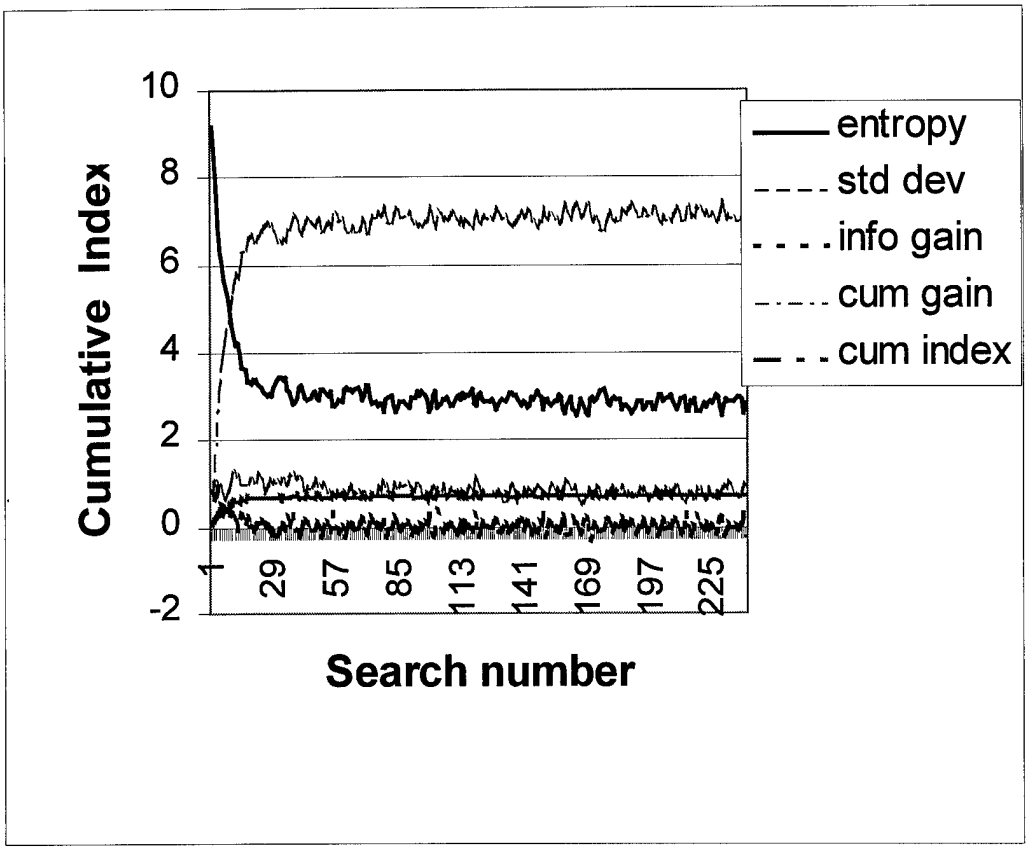


Figure 2. Plot of values for 240 searches corresponding to the parameter settings shown in Table 1. Note the stability of the cumulative gain index (gray curve), at about 0.66, after only about three sweeps through the set of 10 cells.

If the user specifies 1-column output, each run generates an ns-dimensional column of values of the cumulative information gain index at each search. The cumulative information gain index is a convenient measure for comparing results from different runs, because its values are constrained to the interval [-1, 1], which may be interpreted as ($1/100^{th}$ of) the percentage cumulative gain relative to the initial entropy value. This facilitates comparisons when runs have different numbers of targets or different prior distributions, making the initial entropy values different.

If the user does not wish to make additional runs, he or she should click "Stop!" to close the Excel file and stop the program. If the user chooses to make additional runs (by clicking "Another!" in the VB form), he or she is given the opportunity to change parameters in the form, then re-run the simulation. In this case, the columns of output in the Excel Sheet1 are offset to the right, starting at Column G for the first re-run, at

Column N for the second re-run, etc., for 5-column output. The values of the parameters for each run are also written just above the first column of each set of columns containing the output for that run. In this way, one can compare information gain statistics obtained with variations in the values of the input parameters. We used this method to generate the plots of average cumulative information index versus search number shown in the next section of this report.

When the program is run, the parameter-input form is displayed with default values for the parameters (Figure 1). The user may tab to any parameter text box (or click on the box) and change the value shown to any other feasible value. Here, "feasible" refers to requirements that probability inputs be real numbers between 0 and 1, the number of cells be an integer greater than 1, etc. We have included in the program only very basic error checking of user inputs, so it is advisable to input parameter values carefully. If an error of input is detected, the program prompts checking the offending parameter value, then continuing with the run without restarting the program.

The run time for the simulation is roughly proportional to the number of iterations (iter) and the number of searches per iteration (ns); it increase exponentially with the number of targets (nt) and is less sensitive to the number of cells (nc). A "number of iterations completed" picture box in the VB input form shows the user how the simulation is progressing; this may be useful with runs having relatively large values of ns and nt. With a Pentium II processor running at 230 MHz we found the following example run times (where P(tgt moves) = .1, P(detect) = .9, P(false alarm) = .05 and Prior = uniform):

with iter = 20, ns = 100, nt = 3 and nc = 12, the run time was 1/6 minute;
with iter = 20, ns = 100, nt = 3 and nc = 120, the run time was 4/6 minute;
with iter = 20, ns = 100, nt = 30 and nc = 120, the run time was 17 minutes;
with iter = 20, ns = 1000, nt = 30 and nc = 120, the run time was 175 minutes.

If the compiled version is used (running the .exe program), run times are about 40% shorter.

Because of the stochastic nature of the outputs for each iteration, we recommend at least 20 iterations be performed in each run, if time permits (see Figure 3 below). The number of searches need not be larger than that required for the measures of interest to reach "steady state." This will depend on values of parameters such as P(tgt moves) and P(false alarm), and some experimentation may be necessary to determine an appropriate

value for a given set of input parameters. In Figure 3 we show plots of cumulative information index for 5 runs; two with iter = 2, two with iter = 20 and one with iter = 50. The effect of increasing sample size (iter) on stabilizing the cumulative information index is evident in the figure. Parameter values for these runs are shown in Table 2.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.05	0.05	0.05	0.05	0.05	0.05
	Prior	0.1	0.1	0.1	0.1	0.1	0.1
iter=2	2	20	20	50			
ns = 240	240	240	240	240			
nt = 3	3	3	3	3			
nc = 10	10	10	10	10			
P(Mv)0.05	0.05	0.05	0.05	0.05			
dc = 1	1	1	1	1			
run# 1	2	3	4	5			

Table 2. Input for runs plotted in Figure 3.

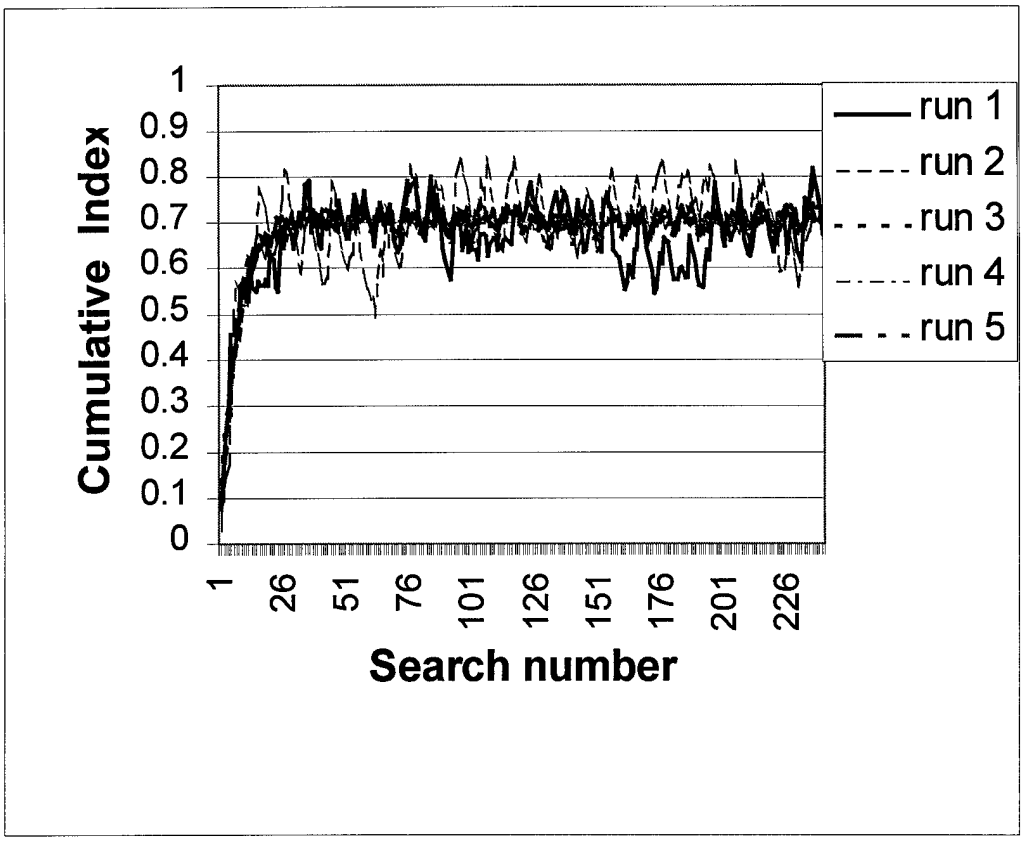


Figure 3. Effects of increasing sample size (iter). Runs 1 (dark curve) and 2 had iter = 2, runs 3 and 4 had iter = 20, and run 5 (gray curve) had iter = 50.

3. ANALYSES OF DATA FROM SELECTED RUNS

We have conducted runs with a number of variations in selected parameters, to provide insights into the effects of such variations. In this section, we present summaries of the results in plots of cumulative information index, usually in five 1-column output runs. The parameters for each run are shown in a table preceding the figure. The tables may truncate the vectors P(detect), P(false), and Prior; when these vectors have constant components the reader can determine the missing values. Brief commentary on each figure is given below the figure.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	Prior	0.055556	0.111111	0.111111	0.1	0.1	0.1	0.11	0.11	0.11	0.06
iter=20	20	20									
ns = 100	100	100									
nt = 3	3	3									
nc = 10	10	10									
P(Mv)0.05	0.05	0.05									
dc = 1	1	1									
run# 1	2	3									

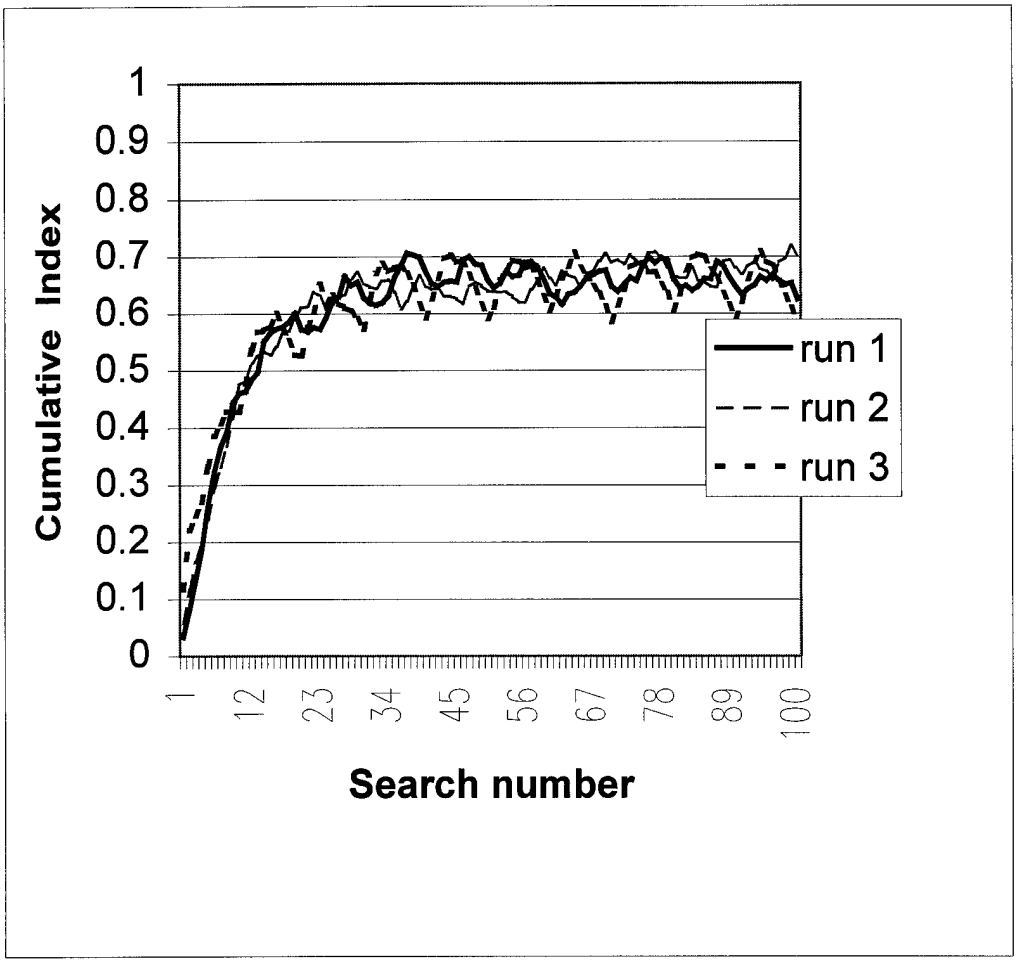


Figure 4. Effects on cumulative information index of changing prior distribution.

Run 1 was made with the limit distribution, run 2 with a uniform prior, and run 3 with a custom prior (values shown in the table above). Priors do not appear to have much effect, after one sweep of the cells, unless target movement probability is very low.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	Prior	0.05	0.05	0.05	0.05	0.05	0.05	0.05
iter=25	25	25	25	25				
ns = 240	240	240	240	240				
nt = 5	5	5	5	5				
nc = 20	20	20	20	20				
P(Mv)0.05	0.05	0.05	0.05	0.05				
dc = 1	2	4	8	12				
run# 1	2	3	4	5				

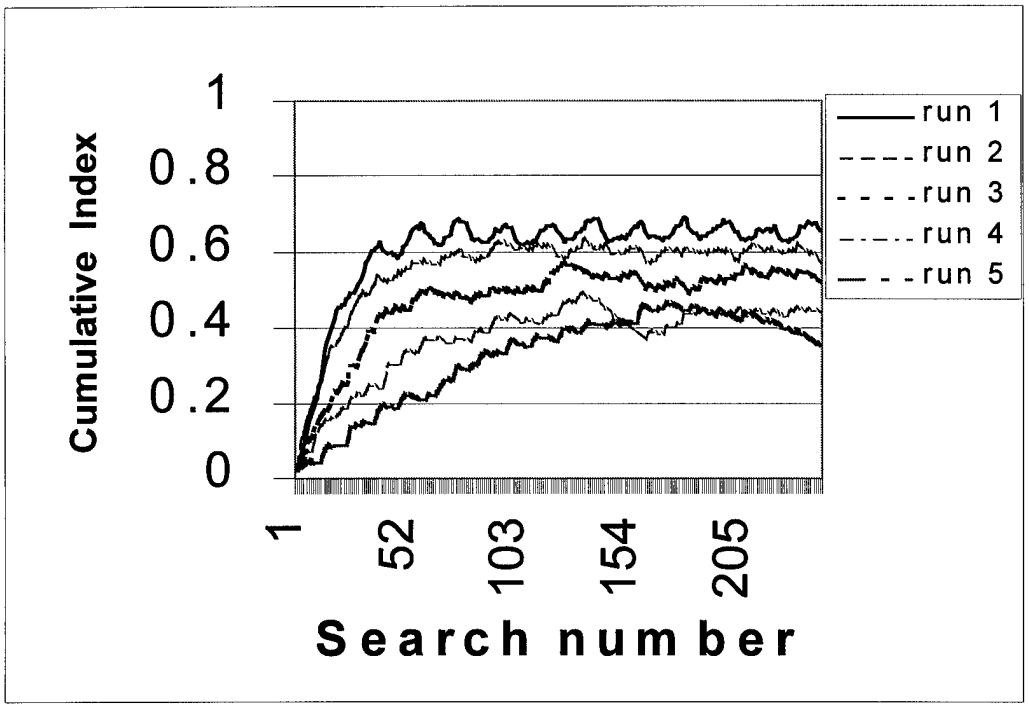


Figure 5. Effects of changing dwell time when target movement is low.

Information accumulates best when there is no dwell (i.e., dc = 1), when target movement probability is low.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	Prior	0.05	0.05	0.05	0.05	0.05	0.05	0.05
iter=25	25	25	25	25				
ns = 240	240	240	240	240				
nt = 5	5	5	5	5				
nc = 20	20	20	20	20				
P(Mv)0.3	0.3	0.3	0.3	0.3				
dc = 1	2	4	8	12				
run# 1	2	3	4	5				

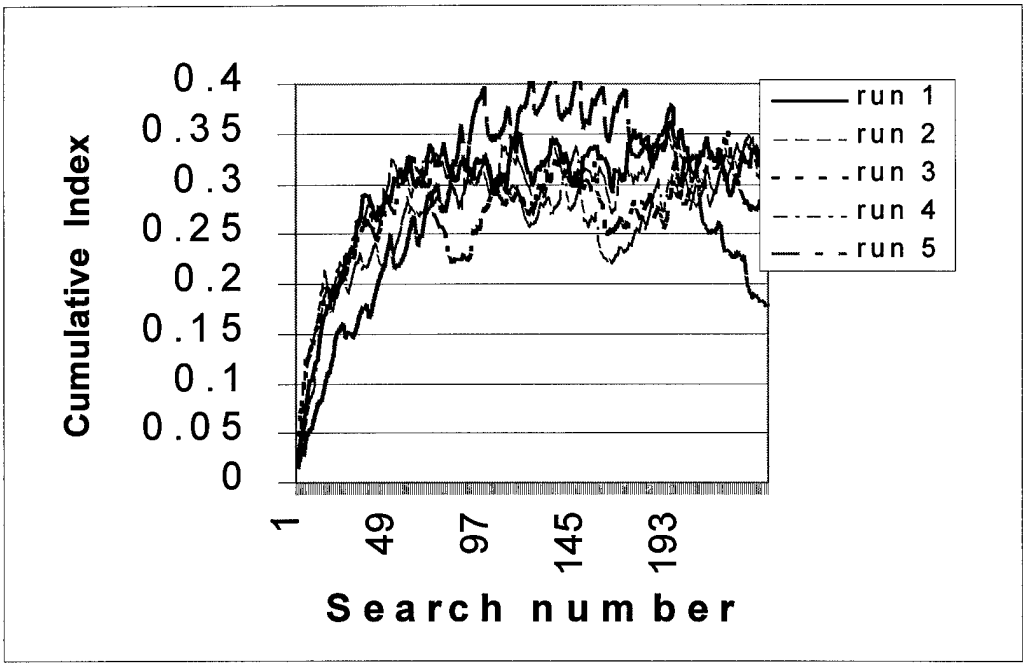


Figure 6. Effects of dwell time when target movement is large.

Overall, there seems to be a slight advantage to larger dwell times when the probability of target movement is large (.3). Contrast this with the situation in Figure 5.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	Prior	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
iter=25	25	25	25	25					
ns = 240	240	240	240	240					
nt = 5	5	5	5	5					
nc = 20	20	20	20	20					
P(Mv)0.5	0.5	0.5	0.5	0.5					
dc = 1	2	4	8	12					
run# 1	2	3	4	5					

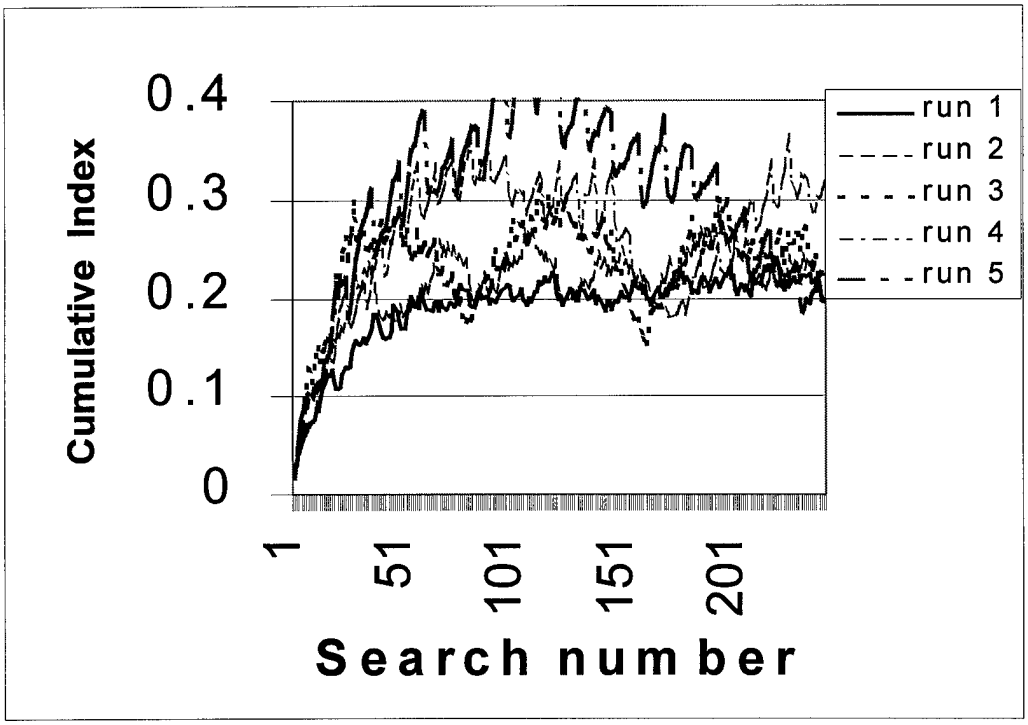


Figure 7. Effects of dwell time when targets are very mobile.

When probability of target movement is very large (.5), larger dwell times appear to give better cumulative information gain. Compare with Figures 5 and 6.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	Prior	0.1	0.1	0.1	0.1	0.1	0.1	0.1
iter=25	25	25	25	25				
ns = 240	240	240	240	240				
nt = 10	10	10	10	10				
nc = 10	10	10	10	10				
P(Mv)0.3	0.3	0.3	0.3	0.3				
dc = 1	2	4	8	12				
run# 1	2	3	4	5				

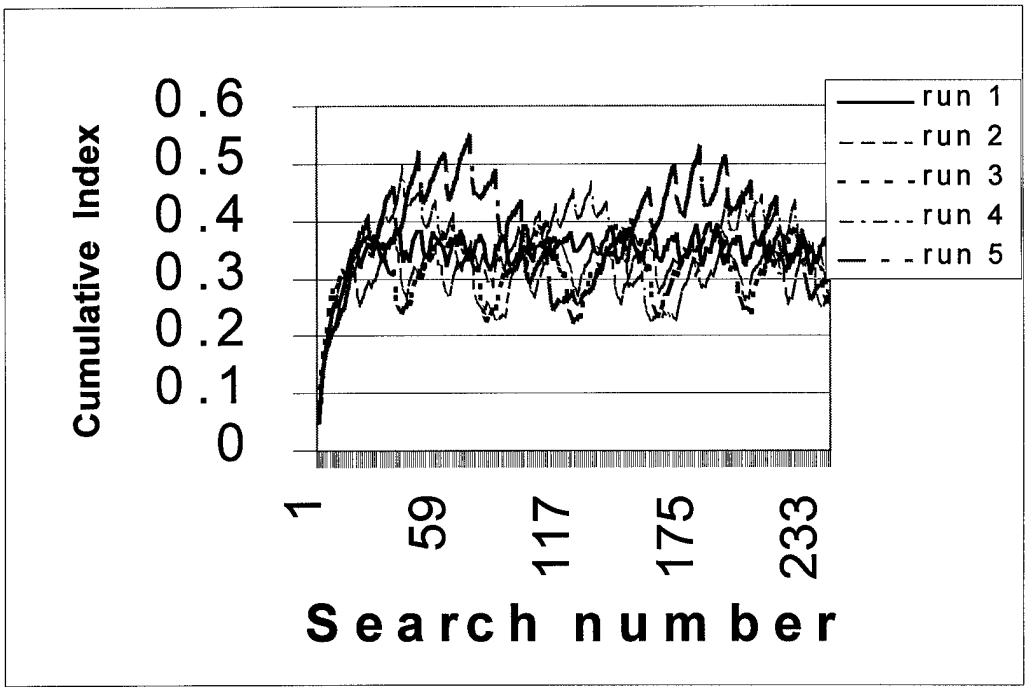


Figure 8. Effects of dwell times in a field rich with very mobile targets.

When there are many targets, relative to the number of cells, and probability of target movement is 0.3, systems with larger dwell times appear to give better cumulative information gain. Compare with Figure 6. The "cyclic" appearance of the curves is due to end-point effects related to the assumed search pattern and reflecting barriers for the target random walk. Note the period of the cycles is equal to the time of a complete sweep through the cells. For example, the curve for dc = 12 (gray curve) has a period of 12(nc) = 120 in this case.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	Prior	0.05	0.05	0.05	0.05	0.05	0.05	0.05
iter=25	25	25	25	25				
ns = 240	240	240	240	240				
nt = 1	1	1	1	1				
nc = 20	20	20	20	20				
P(Mv)0.3	0.3	0.3	0.3	0.3				
dc = 1	2	4	8	12				
run# 1	2	3	4	5				

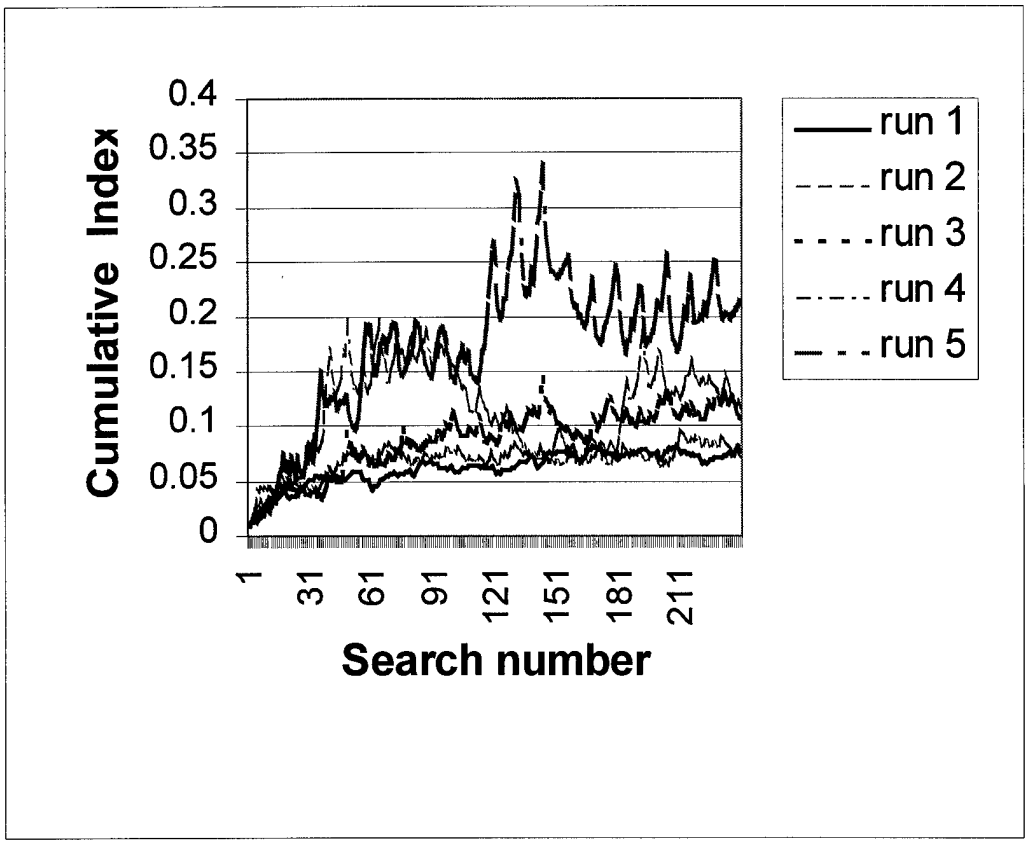


Figure 9. Effect of dwell times in a target poor environment, with P(Mv) = 0.3 and P(false) = 0.4.

In a target poor environment, with poor sensor performance and very mobile targets, larger dwell times appear to give relatively superior cumulative information gains. Compare with Figure 8.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.4	0.4	0.4	0.4	0.4	0.4
	Prior	0.1	0.1	0.1	0.1	0.1	0.1
iter=25	25	25	25	25			
ns = 240	240	240	240	240			
nt = 5	5	5	5	5			
nc = 10	10	10	10	10			
P(Mv)0.1	0.1	0.1	0.1	0.1			
dc = 1	2	4	8	12			
run# 1	2	3	4	5			

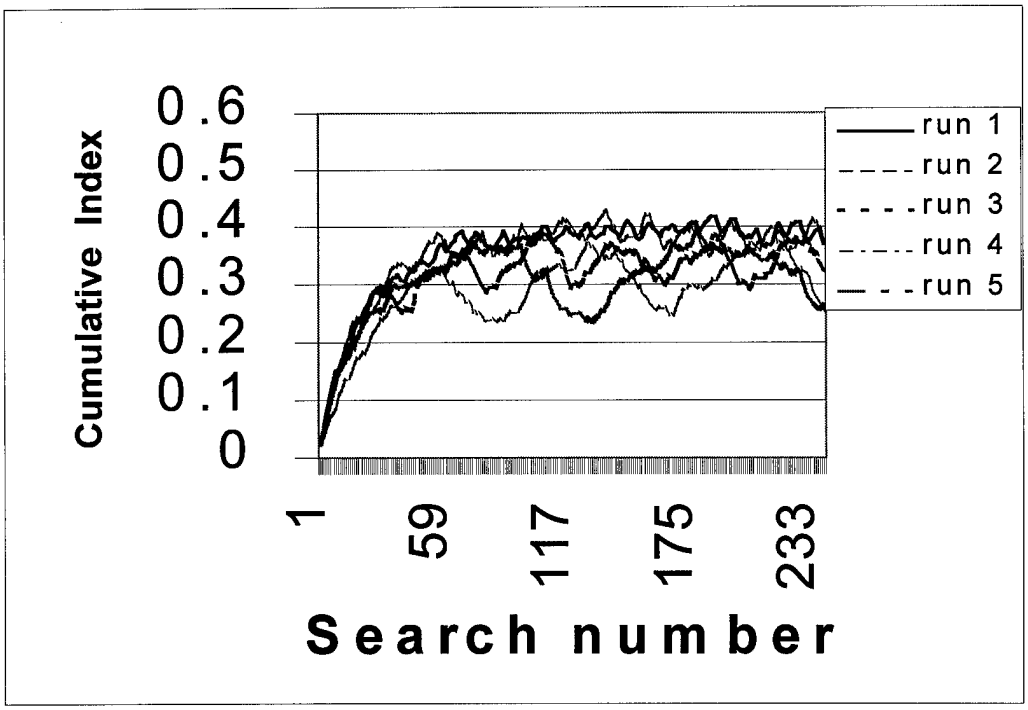


Figure 10. Effects of dwell times with high false alarm probability.

Cumulative information gain has a low steady state value when false alarm rates are high (0.4).

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0	0	0	0	0	0
	Prior	0.1	0.1	0.1	0.1	0.1	0.1
iter=25	25	25	25	25			
ns = 240	240	240	240	240			
nt = 5	5	5	5	5			
nc = 10	10	10	10	10			
P(Mv)0.1	0.1	0.1	0.1	0.1			
dc = 1	2	4	8	12			
run# 1	2	3	4	5			

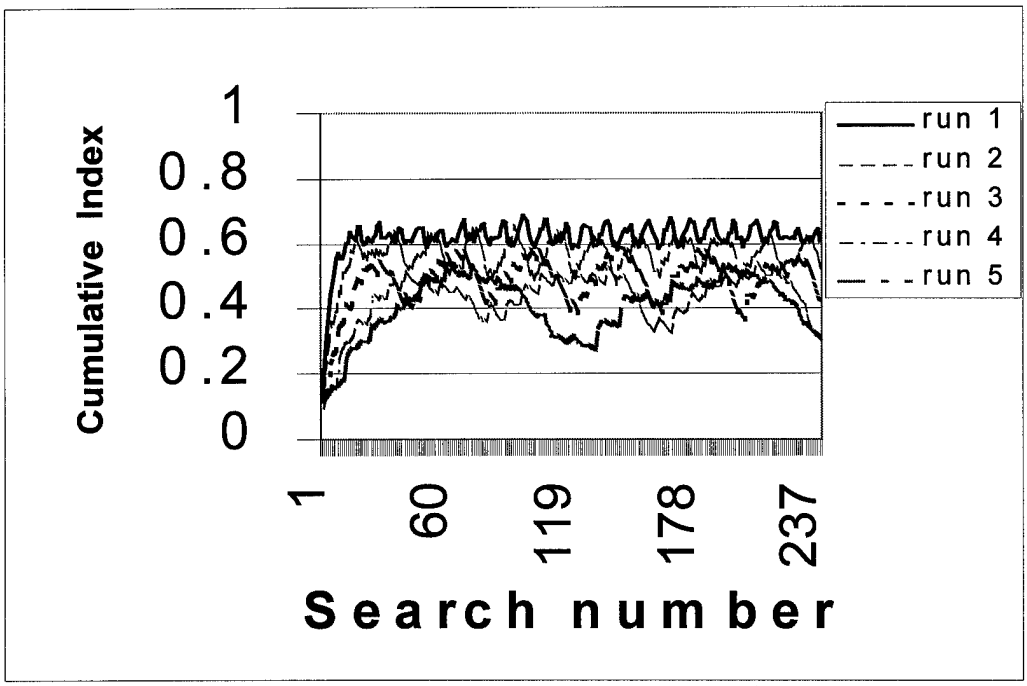


Figure 11. Effects of dwell times when there are no false alarms.

Short dwell times appear best when the sensor cannot report false detection. Compare with Figure 10.

	P(detect)	0.9	0.9	0.9	0.9	0.9	0.9
	P(false)	0.05	0.05	0.05	0.05	0.05	0.05
	Prior	0.1	0.1	0.1	0.1	0.1	0.1
iter=25	25	25	25	25			
ns = 240	240	240	240	240			
nt = 3	3	3	3	3			
nc = 10	10	10	10	10			
P(Mv)0	0	0	0	0			
dc = 1	2	4	8	12			
run# 1	2	3	4	5			

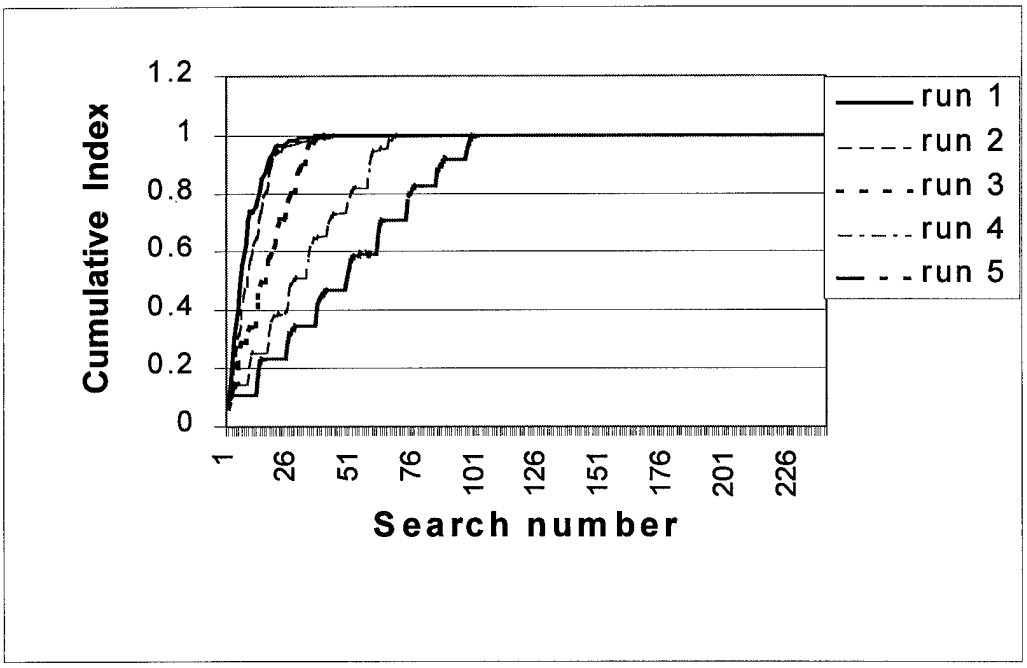


Figure 12. Effect of dwell times with immobile targets.

In case targets cannot move, increased dwell times appear to be of no value, in terms of cumulative information gain.

Conclusions

The simple simulation we have developed can give insight into possible tradeoffs between dwell time and frequency of visit of a sensor. In our analysis, frequency of visit is constrained to simple multiples of dwell time, so the latter might more properly be called "dwell ratio." For example, when dwell time is set to 2, the dwelling sensor glimpses in each cell twice before moving to the adjacent cell, whereas the non-dwelling sensor moves after each glimpse. Therefore, the frequency of visit to a specific cell with the non-dwelling sensor is twice that of this dwelling sensor.

It appears a dwelling sensor may have advantage when targets are very mobile. This is because the dwelling sensor is assumed to track a detected target during the dwell period, and such a target does not have information decaying because of movement. It is possible the beneficial effect of larger dwell in such cases is tracking ability, rather than dwell itself. If dwell times are too large, the performance of the sensor system is degraded. Here, "large" is measured relative to the number of cells in the area of concern; it also appears to depend on target mobility. If targets are relatively immobile and dwell times are very large, then it may take a long time for the sensor system to reach a cell occupied by a target, whereas the non-dwell sensor will cycle past the target more frequently. In addition, the tracking advantage of the dwelling sensor is degraded with a target that rarely (or never) moves.

The documentation and user instructions we have supplied should allow others to perform experiments with the simulation we have developed. A listing of the Visual Basic code is given in the Appendix. The compiled version of the program does not require Visual Basic on the running machine, although Excel '97 is required.

REFERENCES

Barr, D. and T. Sherrill. 1995. Estimating the Operational Value of Tactical Information. Operations Research Center Technical Report, U.S. Military Academy, West Point, NY.

Barr, D., and T. Sherrill. 1996. Measuring Information Gain in Tactical Operations. Operations Research Center Technical Report, U.S. Military Academy, West Point, NY.

Goodman, R. 1988. Introduction to Stochastic Models. The Benjamin/Cummings Publishing Co., Menlo Park.

Marin, J. and D. Barr. 1997. Evaluation of Intelligent Minefields. *Military Operations Research* **3**, 47-60.

MacWillie, S. 1992. Contributions of Reconnaissance: MOE/MOP Guide. Directorate of Combat Developments, US Army Aviation Center Combat Development Pamphlet, Fort Rucker, AL.

Shannon, C. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal* **27**, 379-423.

Sherrill, T. and D. Barr. 1996. Exploring a Relationship Between Tactical Intelligence and Battle Results. *Military Operations Research* **2**, 17-33.

Sherrill, T., and D. Barr. 1997. Assessing Situation Awareness in Task Force XXI. Operations Research Center Technical Report, U.S. Military Academy, West Point, NY.

Sovereign, M. 1996. Warfare in the Information Age: Vision 2010 and Changes in Joint C^4 Doctrine. Proceedings of the 1996 Command & Control Research & Technology Symposium, National Defense University Report, D. Alberts, editor.

TRADOC Pamphlet 525-5. 1994. "Force XXI Operations," United States Army Training and Doctrine Command, Fort Monroe, VA.

APPENDIX: LISTING OF VISUAL BASIC CODE

A. General Module:

```
Private iter As Integer              'number of iterations
Private ns As Integer                'number of searches per iteration
Private nt As Integer                'number of targets
Private nc As Integer                'number of cells
Private mp As Single                 'probability target moves in each cycle
Private q As Double                  'non-move prob
Private dc As Integer                'number of dwell cycles per cell
Private Li As Integer                'iteration counter
Private K As Integer                 'search counter
Dim pd() As Single                   'probability of detection, given target present
Dim pf() As Single                   'probability of false alarm
Dim L() As Integer                   'locations of targets
Dim LOS() As Integer                 'flag for LOS with detected target
Dim P() As Double                    'location probabilities for targets
Dim CP() As Single                   'cdf of the prior
Dim PW() As Double                   'nt x nc working array for decay
Dim Ent() As Double                  'entropy accumulator
Dim Ents() As Double                 'sum of squares counter
Dim IGainAv() As Single              'information gain at Kth search
Dim CGainAV() As Single              'cumulative gain at Kth search
Dim CGainIdx() As Single             'cumulative gain index
Dim I As Integer                     'target index
Dim PointC As Integer                'cell to be searched
Dim Relook As Integer                'counter for dwell cycles
Dim avpd As Single                   'average detection prob
Dim xlio As Excel.Workbook           'get ready to open Excel file
Dim OutCol As Integer                'count of number of runs - 1
Dim numcol As Integer                'output columns multiplier
Dim msg$                             'error message
```

B. Subroutines

```
Private Sub Command1_Click()
InitializeO                          'overall initialize: read pd, pf, prior
    If msg$ <> "" Then Exit Sub      'stop if input not OK

For Li = 1 To iter                   'start main driver
    InitializeI                      'initialize iteration: zero D, F, set up L, P
    For K = 1 To ns                  'search K in iteration Li
        Pointer                      'cell to search
        DecayP                       'modify P to account for movement probabilities
        Search                       'search cell, update D, F
        Mover                        'stochastic movement of targets
        Entropy                      'compute entropy, update counters
    Next K
        DisplayIt
Next Li

Output                               'compute stats, write to file

End Sub                              'return to input form
```

```
Sub Initialize0()                    'do initial housekeeping, input parameters
    msg$ = ""                        'error message
    If OutCol = 0 Then               'first time, open file
     Set xlio = GetObject(Text8.Text) 'excel input-output file
    End If
    Label9.Caption = ""              'take message off screen
    Text9.Text = ""                  'hide any message
    Picture1.Cls
    Picture1.Print "0"               'iterations counter
    Picture1.SetFocus                'now set up Excel Sheet1 headers
    xlio.Worksheets("Sheet1").Cells(1, 1) = "DWELL TIME vs.REVISIT FREQUENCY"
    xlio.Worksheets("Sheet1").Cells(3, 2) = "P(detect)"
    xlio.Worksheets("Sheet1").Cells(4, 2) = "P(false)"
    xlio.Worksheets("Sheet1").Cells(5, 2) = "Prior"

    If Option2.Value = True Then
      numcol = 5                     'set # output columns flag
    Else: numcol = 1
    End If
    X$ = ""
 iterx = CSng(Text1.Text)            'write user inputs to Excel Sheet1
    If iterx < 1 Or Int(iterx) <> iterx Then
        msg$ = "check iter"
        errmsg
    Else
        iter = CInt(iterx)
    End If
    If OutCol = 0 Then X$ = "iter="
    xlio.Worksheets("Sheet1").Cells(6, 1 + numcol * OutCol) = X$ & CStr(iter)
 nsx = CSng(Text2.Text)
    If nsx < 2 Or Int(nsx) <> nsx Then
        msg$ = "check ns"
        errmsg
    Else
        ns = CInt(nsx)
    End If
    If OutCol = 0 Then X$ = "ns = "
    xlio.Worksheets("Sheet1").Cells(7, 1 + numcol * OutCol) = X$ & CStr(ns)
 ntx = CSng(Text3.Text)
    If ntx < 1 Or Int(ntx) <> ntx Then
        msg$ = "check nt"
        errmsg
    Else
        nt = CInt(ntx)
    End If
    If OutCol = 0 Then X$ = "nt = "
    xlio.Worksheets("Sheet1").Cells(8, 1 + numcol * OutCol) = X$ & CStr(nt)
 ncx = CSng(Text4.Text)
    If ncx < 2 Or Int(ntx) <> ntx Then
        msg$ = "check nc"
        errmsg
    Else
        nc = CInt(ncx)
```

```
    End If
    If OutCol = 0 Then X$ = "nc = "
    xlio.Worksheets("Sheet1").Cells(9, 1 + numcol * OutCol) = X$ & CStr(nc)
q = 1 - CDbl(Text6.Text)            'prob tgt doesn't move
    If q < 0 Or q > 1 Then
        msg$ = "check move prob"
        errmsg
    End If
    If OutCol = 0 Then X$ = "P(Mv)"
    xlio.Worksheets("Sheet1").Cells(10, 1 + numcol * OutCol) = X$ & CStr(1 - q)
dcx = CSng(Text5.Text)
    If dcx < 1 Or Int(dcx) <> dcx Then
        msg$ = "check dc"
        errmsg
    Else
        dc = CInt(dcx)
    End If
    If Text11.Text <> "" And (CSng(Text11.Text) < 0 Or CSng(Text11.Text) > 1) Then
        msg$ = "check P(detect)"
        errmsg
    End If
    If Text12.Text <> "" And (CSng(Text12.Text) < 0 Or CSng(Text12.Text) > 1) Then
        msg$ = "check P(F. Alrm)"
        errmsg
    End If
    If msg$ <> "" Then Exit Sub
    If OutCol = 0 Then X$ = "dc = "
    xlio.Worksheets("Sheet1").Cells(11, 1 + numcol * OutCol) = X$ & CStr(dc)
    If OutCol = 0 Then X$ = "run# "
    xlio.Worksheets("Sheet1").Cells(12, 1 + numcol * OutCol) = X$ & CStr(OutCol + 1)

    Randomize                       'seed random number generator
    ReDim PW(nt, nc - 1)            'dynamic arrays
    ReDim pd(nc - 1)      'cell counter range is 0 to nc-1, for modular arithmetic
    ReDim pf(nc - 1)
    ReDim L(nt, nc - 1)
    ReDim LOS(nt, nc - 1)
    ReDim P(nt, nc - 1)             'P(0,.) retains copy of prior
    ReDim CP(-1 To nc - 1)          'CDF of custom prior, used to generate locations
    ReDim Ent(ns)
    ReDim Ents(ns)
    ReDim IGainAv(ns)
    ReDim CGainAV(ns)
    ReDim CGainIdx(ns)

    If Text11.Text = "" Then        'read pd from Excel Sheet1
        avpd = 0                    'counter for average pd
      For j = 0 To nc - 1           'read pd
        pd(j) = xlio.Worksheets("Sheet1").Cells(3, 3 + j)
        If pd(j) < 0 Or pd(j) > 1 Then
          msg$ = " check pd on Sheet1"
          errmsg
        End If
        avpd = avpd + pd(j)
```

```
  Next j
    avpd = avpd / nc              'average pd for false alarms in Search
Else                              'set pd from Form1, write to Excel Sheet1
  avpd = CSng(Text11.Text)
  For j = 0 To nc - 1
    pd(j) = avpd
    xlio.Worksheets("Sheet1").Cells(3, 3 + j) = pd(j)
  Next j
End If
If Text12.Text = "" Then          'read pf from Excel Sheet1
  For j = 1 To nc - 1
    pf(j) = xlio.Worksheets("Sheet1").Cells(4, 3 + j)
    If pf(j) < 0 Or pf(j) > 1 Then
      msg$ = " check pf on Sheet1"
      errmsg
    End If
  Next j
Else                              'read pf from VB Form1, write to Excel Sheet1
  For j = 0 To nc - 1
    pf(j) = CSng(Text12.Text)
    xlio.Worksheets("Sheet1").Cells(4, 3 + j) = pf(j)
  Next j
End If
  If Text13.Text = "uniform" Then   'set uniform prior
    Ent(0) = nt * Log(nc) / Log(2)  'initial entropy
    For j = 0 To nc - 1
      P(0, j) = 1 / nc
      xlio.Worksheets("Sheet1").Cells(5, 3 + j) = P(0, j)
    Next j
  ElseIf Text13.Text = "limit" Then 'set prior for random walk
    Ent(0) = nt * (1 / (nc - 1) + Log(nc - 1) / Log(2))  'initial entropy
    For j = 0 To nc - 1
      P(0, j) = 1 / (nc - 1)
      If j = 0 Then P(0, j) = 1 / (2 * (nc - 1))
      If j = nc - 1 Then P(0, nc - 1) = 1 / (2 * (nc - 1))
      xlio.Worksheets("Sheet1").Cells(5, 3 + j) = P(0, j)
    Next j
  Else                          'read prior from Excel Sheet1, set initial entropy
      Ent(0) = 0
    For j = 0 To nc - 1
      P(0, j) = xlio.Worksheets("Sheet1").Cells(5, 3 + j) 'read prior
        If P(0, j) < 0 Or P(0, j) > 1 Then                    'from Excel Sheet1
          msg$ = "check prior on Sheet 1"
          errmsg
        End If
          If P(0, j) > 0.0000000001 Then Ent(0) = Ent(0) - P(0, j) * Log(P(0, j))
   CP(j) = CP(j - 1) + P(0, j)       'CDF to generate initial locations
    Next j
       If CP(nc - 1) < 0.99999 Or CP(nc - 1) > 1.00001 Then
         msg$ = " check prior on Sheet1"
         errmsg
       End If
    Ent(0) = nt * Ent(0) / Log(2)  'total entropy, base 2
  End If
```

```
End Sub

Sub InitializeI()                    'zero LOS; draw L; set Priors
    For I = 1 To nt
        For j = 0 To nc - 1
            LOS(I, j) = 0            'start with no LOS
            L(I, j) = 0              'prepare to set locations
            P(I, j) = P(0, j)        'set priors
        Next j
    Next I
        If Text13.Text = "uniform" Then    'draw locations
          For I = 1 To nt
            L(I, Int(nc * Rnd)) = 1        'random initial locations
          Next I
        ElseIf Text13.Text = "limit" Then  'limiting distribution for random walk
          For I = 1 To nt                  'set prior values
            L(I, Int((nc - 1) * Rnd)) = 1  'draw initial locations
             If L(I, 0) = 1 And Rnd < 0.5 Then
               L(I, 0) = 0
               L(I, nc - 1) = 1
             End If
          Next I
        Else                             'custom prior from Excel Sheet1
            For I = 1 To nt              'now draw L
               xx = Rnd
             For j = 0 To nc - 1
               If xx <= CP(j) Then Exit For  'inverse CDF generation of location
             Next j
             L(I, j) = 1
           Next I
           If Text13.Text = "uniform" Or Text13.Text = "limit" Then
             For j = 0 To nc - 1     'write prior to Excel Sheet1
               xlio.Worksheets("Sheet1").Cells(5, 3 + j) = P(1, j)
             Next j
           End If
        End If
End Sub

Sub Pointer()                        'set cell pointer for search
    If K = 1 Then PointC = 0: Relook = 1: Exit Sub   'start in cell 0
    Relook = Relook + 1              'dwell counter
    If Relook > dc Then              'accomodate dwell time dc
        PointC = PointC + 1
        If PointC > nc - 1 Then PointC = 0 'cyclic search pattern
        Relook = 1                   'restart dwell counter
        For I = 1 To nt              'LOS to old cell is lost
          LOS(I, (PointC - 1 + nc) Mod nc) = 0
        Next I
    End If
End Sub

Sub DecayP()                         'decay using known random movement probs
    If K = 1 Then Exit Sub           'lag decay one cycle
    For I = 1 To nt
```

```
      If LOS(I, PointC) = 0 Then    'no decay when in view, else decay P

        For I1 = 1 To nt
          For J1 = 0 To nc - 1
            PW(I1, J1) = P(I1, J1)  'copy current P
          Next J1
        Next I1

    Select Case nc                  'random walk eqns
      Case 2
        P(I, 0) = q * PW(I, 0) + (1 - q) * PW(I, 1)
        P(I, 1) = (1 - q) * PW(I, 0) + q * PW(I, 1)
      Case 3
        P(I, 0) = q * PW(I, 0) + ((1 - q) / 2) * PW(I, 1)
        P(I, 1) = (1 - q) * PW(I, 0) + q * PW(I, 1) + (1 - q) * PW(I, 2)
        P(I, 2) = ((1 - q) / 2) * PW(I, 1) + q * PW(I, 2)
      Case Is >= 4
        P(I, 0) = q * PW(I, 0) + ((1 - q) / 2) * PW(I, 1)
        P(I, 1) = (1 - q) * PW(I, 0) + q * PW(I, 1) + ((1 - q) / 2) * PW(I, 2)
        For j = 2 To nc - 3
 P(I, j) = ((1 - q) / 2) * PW(I, j - 1) + q * PW(I, j) + ((1 - q) / 2) * PW(I, j + 1)
        Next j
 P(I, nc - 2) = ((1 - q) / 2) * PW(I, nc - 3) + q * PW(I, nc - 2) +  _
                                    (1 - q) * PW(I, nc - 1)
    P(I, nc - 1) = ((1 - q) / 2) * PW(I, nc - 2) + q * PW(I, nc - 1)
    End Select
    End If
  Next I
End Sub

Sub Search()                        'search PointC, update D, F, P
    For I = 1 To nt
      If L(I, PointC) = 1 Then      'looking at target
          If Rnd < pd(PointC) Then  'detection!
            LOS(I, PointC) = 1
            If pf(PointC) = 0 Then P(I, PointC) = 1 'must be real detection
              Updated
          Else
              Updaten               'looking at tgt, but no detect
          End If
      ElseIf Rnd < 1 - (1 - pf(PointC)) ^ (1 / (nt * avpd)) And LOS(I, PointC) = 0
Then
            Updated
      Else
            Updaten                 'no "detection"
      End If
    Next I
End Sub

Sub Updated()                      'update P with "detection" using Bayes' formula
   Sum = 0                          'normalization counter
  For j = 0 To nc - 1
    If j <> PointC Then
      P(I, j) = pf( PointC) * P(I, j)
```

```
      Sum = Sum + P(I, j)
    Else
      P(I, PointC) = pd(PointC) * P(I, PointC)
      Sum = Sum + P(I, PointC)
    End If
  Next j
  For j = 0 To nc - 1                'normalize P(I,.) vector
    P(I, j) = P(I, j) / Sum
  Next j
End Sub

Sub Updaten()                        'update P with no "detection" using Bayes'
formula
   Sum = 0                           'normalization counter
  For j = 0 To nc - 1
    If j <> PointC Then
      P(I, j) = (1 - pf(PointC)) * P(I, j)
      Sum = Sum + P(I, j)
    Else
      P(I, PointC) = (1 - pd(PointC)) * P(I, PointC)
      Sum = Sum + P(I, PointC)
    End If
   Next j
   For j = 0 To nc - 1
     P(I, j) = P(I, j) / Sum         'normalize vector
   Next j
End Sub

Sub Mover()                          'stochastic movement of targets
  For I = 1 To nt
    For j = 0 To nc - 1              'determine where this tgt is
      If L(I, j) = 1 Then Exit For
    Next j
      xx = Rnd                       'move random variable
    If LOS(I, j) = 0 And xx >= q Then 'detected tgts can't move during dwell
                                     '(to model tracking); no move for this target if xx<q
      If j = 0 Then                  'move tgt to the right
        L(I, 0) = 0
        L(I, 1) = 1
      ElseIf j = nc - 1 Then         'move to the left
        L(I, nc - 1) = 0
        L(I, nc - 2) = 1

      ElseIf xx > (1 + q) / 2 Then   'move left or right with prob 1-q
        L(I, j - 1) = 1
        L(I, j) = 0
      Else
        L(I, j + 1) = 1
        L(I, j) = 0
      End If
    End If
 Next I
End Sub
```

```
Sub Entropy()                          'compute entropy and update counters
    Entrop = 0
    For I = 1 To nt                    'cumulative (over L) entropy at search K
      For j = 0 To nc - 1
        If P(I, j) > 0.0000000000001 Then   'avoid underflow problems
          Entrop = Entrop - P(I, j) * Log(P(I, j))  'base e
        End If
      Next j
    Next I
    Entrop = Entrop / Log(2)           'base 2
    Ent(K) = Ent(K) + Entrop           'cumulative sum
    Ents(K) = Ents(K) + Entrop * Entrop 'sum of squares
End Sub

Sub Output()                           'compute and write out results
For K = 1 To ns                        'compute statistics
    Ent(K) = Ent(K) / iter             'average for search K
    If iter > 1 And (Ents(K) - iter * Ent(K) * Ent(K)) > 0 Then
      Ents(K) = Sqr((Ents(K) - iter * Ent(K) * Ent(K)) / CDbl(iter - 1))  'std dev
    Else
      Ents(K) = 0
    End If
    IGainAv(K) = Ent(K - 1) - Ent(K)            'average info gain due to Kth search
    CGainAV(K) = CGainAV(K - 1) + IGainAv(K)    'cumulative gain
    CGainIdx(K) = CGainAV(K) / Ent(0)           'cumulative index
Next K

    If Option1.Value = True Then               'one column output
      xlio.Worksheets("Sheet1").Cells(13, 1 + OutCol) = "Cum. Gain"
      xlio.Worksheets("Sheet1").Cells(14, 1 + OutCol) = "  Index"
      For K = 1 To ns
       xlio.Worksheets("Sheet1").Cells(14 + K, 1 + OutCol) = CGainIdx(K)
      Next K
    Else                               '5 col output
      xlio.Worksheets("Sheet1").Cells(13, 1 + 5 * OutCol) = "Entropy"
      xlio.Worksheets("Sheet1").Cells(14, 1 + 5 * OutCol) = "Mean"
      xlio.Worksheets("Sheet1").Cells(13, 2 + 5 * OutCol) = "Entropy"
      xlio.Worksheets("Sheet1").Cells(14, 2 + 5 * OutCol) = "std dev"
      xlio.Worksheets("Sheet1").Cells(14, 3 + 5 * OutCol) = "Info Gain"
      xlio.Worksheets("Sheet1").Cells(14, 4 + 5 * OutCol) = "Cum Gain"
      xlio.Worksheets("Sheet1").Cells(13, 5 + 5 * OutCol) = "Cum Gain"
      xlio.Worksheets("Sheet1").Cells(14, 5 + 5 * OutCol) = "Index"
      For K = 1 To ns
       xlio.Worksheets("Sheet1").Cells(14 + K, 1 + 5 * OutCol) = Ent(K)
       xlio.Worksheets("Sheet1").Cells(14 + K, 2 + 5 * OutCol) = Ents(K)
       xlio.Worksheets("Sheet1").Cells(14 + K, 3 + 5 * OutCol) = IGainAv(K)
       xlio.Worksheets("Sheet1").Cells(14 + K, 4 + 5 * OutCol) = CGainAV(K)
       xlio.Worksheets("Sheet1").Cells(14 + K, 5 + 5 * OutCol) = CGainIdx(K)
      Next K
    End If
    Text9.Text = "Another!"            'display option for another run
    Text9.SetFocus

' *** write-out of matrices to Sheet 2***
```

```
'For I = 1 To nt                        'write rows of L, D, F, P, filling rest with
zeros
 ' For J = 0 To nc - 1
  '  xlio.Worksheets("Sheet2").Cells(I + 11, J + 1) = L(I, J)
   ' xlio.Worksheets("Sheet2").Cells(I + 21, J + 1) = LOS(I, J)
    'xlio.Worksheets("Sheet2").Cells(I + 1, J + 1) = P(I, J)
 ' Next J
'Next I
' ***************

End Sub

Private Sub Text10_Click()
  xlio.Save                             'save output to file
  Set xlio = Nothing                    'release object
  End
End Sub

Private Sub Text9_Click()               'user gets option to make another run
    Label9.Caption = "set parameters, click 'go'"
    OutCol = OutCol + 1
    Picture1.Cls
    Picture1.Print "0"
    Command1.SetFocus
End Sub

Sub DisplayIt()                         'display iterations completed
   Picture1.Cls
   Picture1.Print CStr(Li);
   Picture1.SetFocus
End Sub

Sub errmsg()
  If Left$(msg$, 1) = " " Then
    Label9.Caption = msg$ & "; fix, start over"
  Else
    Label9.Caption = msg$ & "; fix, click 'go'"
  End If
    Text9.Text = "ERROR!"
 End Sub
```